MC102 - Algoritmos e Progração de Computador

Prof. Alexandre Xavier Falcão

21° Aula: Alocação dinâmica de memória (cont.)

1 Alocando memória para matrizes

No caso de vetores multidimensionais (e.g. matrizes), devemos alocar um vetor de apontadores, i.e. um apontador por linha, e depois um vetor de elementos para cada linha. Assim como fizemos para Figura, podemos entender uma matriz como uma estrutura abstrata com operações de criação, destruição, transposição, leitura e impressão. O programa abaixo ilustra esses conceitos para o caso de matriz representada por um registro. Fica como exercício modificá-lo para representar a matriz como apontador para registro, assim como fizemos na aula de alocação de memória para vetores.

```
#include <stdio.h>
#include <malloc.h>

typedef struct _matriz {
  float **elem; /* matriz de elementos do tipo float */
  int nlin,ncol; /* números de linhas e colunas */
} Matriz;

Matriz CriaMatriz(int nlin, int ncol);
Matriz LeMatriz();
Matriz TranspoeMatriz(Matriz m1);
void   ImprimeMatriz(Matriz m);
void   DestroiMatriz(Matriz m);

Matriz CriaMatriz(int nlin, int ncol)
{
  Matriz m;
  int l;

  m.elem = (float **)calloc(nlin,sizeof(float *)); /* aloca vetor de
                                                      apontadores para
                                                      variáveis do
                                                      tipo float */
  if (m.elem != NULL){
    for (l=0; l < nlin; l++)
      m.elem[l] = (float *)calloc(ncol,sizeof(float)); /* aloca vetor
                                                          de variáveis
```

```
                                                    do tipo
                                                    float */

    m.nlin = nlin;
    m.ncol = ncol;
  }

  return(m); /* retorna cópia do registro */
}

Matriz  LeMatriz()
{
  int l,c,nlin,ncol;
  Matriz m;

  scanf("%d %d",&nlin,&ncol);
  m = CriaMatriz(nlin,ncol);
  for (l=0; l < m.nlin; l++)
    for (c=0; c < m.ncol; c++)
      scanf("%f",&m.elem[l][c]);
  return(m);
}

Matriz TranspoeMatriz(Matriz m1)
{
  int l,c;
  Matriz m2;

  m2 = CriaMatriz(m1.ncol,m1.nlin);
  for (l=0; l < m2.nlin; l++)
    for (c=0; c < m2.ncol; c++)
      m2.elem[l][c] = m1.elem[c][l];
  return(m2);
}

void   ImprimeMatriz(Matriz m)
{
  int l,c;

  for (l=0; l < m.nlin; l++){
    for (c=0; c < m.ncol; c++)
      printf("%f ",m.elem[l][c]);
    printf("\n");
  }
}
```

```
void DestroiMatriz(Matriz m)
{
  int l;

  if (m.elem != NULL) {
    for (l=0; l < m.nlin; l++) /* desaloca espaço do vetor de
                                  variáveis de cada linha */
      if (m.elem[l] != NULL)
        free(m.elem[l]);
    free(m.elem); /* desaloca espaço do vetor de apontadores */
  }
}

int main()
{
  Matriz m1,m2;

  m1 = LeMatriz();
  m2 = TranspoeMatriz(m1);
  ImprimeMatriz(m2);
  DestroiMatriz(m1);
  DestroiMatriz(m2);

  return 0;
}
```

2 Outras formas de manipulação de apontadores duplos

O programa abaixo ilustra outras formas de manipulação de apontadores duplos, que costumam confundir bastante os programadores.

```
#include <malloc.h>

/*----- 1. Manipulação com alocação e desalocação dinâmicas de memória ----*/

float **CriaMatriz(int nlin, int ncol); /* Aloca memória para matriz
                                           representada por apontador
                                           duplo */

void    DestroiMatriz(float ***m, int nlin); /* Desaloca memória
                                                atribuindo NULL ao
                                                conteúdo de m. Isto
                                                requer passar o
                                                endereço de m para a
                                                função, pontanto m é um
                                                apontador de apontador
```

```
                                                duplo. Ou seja,
                                                apontador triplo. */

/*----- 2. Manipulação no caso de alocação estática ----------------------*/

void TraspoeMatriz(float m1[2][3], float m2[3][2]); /* Transpõe m1 para m2 */

/*---------------------- Escopos das funções -----------------------------*/

float **CriaMatriz(int nlin, int ncol)
{
  float **m=NULL;
  int l;

  m = (float **)calloc(nlin,sizeof(float *));
  if (m != NULL)
    for (l=0; l < nlin; l++)
      m[l]=(float *) calloc(ncol,sizeof(float));
  return(m);
}

void DestroiMatriz(float ***m, int nlin)
{
  int l;

  if (**m != NULL) {
    if (*m != NULL) {
      for (l=0; l < nlin; l++)
        free((*m)[l]);
      free(*m);
      *m = NULL;
    }
  }
}
/* ----------------------------------------------------------*/

void TranspoeMatriz(float m1[2][3], float m2[3][2]) /* passagem por
                                                       referência */
{
  int l,c;

  for (l=0; l < 3; l++)
    for (c=0; c < 2; c++)
      m2[l][c] = m1[c][l];
}
```

```
int main()
{
  float **m=NULL;
  float m1[2][3]={{1,2,3},{4,5,6}}; /* inicializa por linha */
  float m2[3][2];
  int l,c;

  m = CriaMatriz(2,3);
  printf("%d\n",m);
  DestroiMatriz(&m,2);
  printf("%d\n",m);
  /* ------------------------- */
  for (l=0; l < 2; l++) {
    for (c=0; c < 3; c++)
      printf("%f ",m1[l][c]);
    printf("\n");
  }
  TranspoeMatriz(m1,m2);
  for (l=0; l < 3; l++) {
    for (c=0; c < 2; c++)
      printf("%f ",m2[l][c]);
    printf("\n");
  }

  return 0;
}
```